

```

;*****
; Copyright © [01/25/1999] Scenix Semiconductor, Inc. All rights reserved.
;
; Scenix Semiconductor, Inc. assumes no responsibility or liability for
; the use of this [product, application, software, any of these products].
; Scenix Semiconductor conveys no license, implicitly or otherwise, under
; any intellectual property rights.
; Information contained in this publication regarding (e.g.: application,
; implementation) and the like is intended through suggestion only and may
; be superseded by updates. Scenix Semiconductor makes no representation
; or warranties with respect to the accuracy or use of these information,
; or infringement of patents arising from such use or otherwise.
;*****
;
; Filename:      simple_fsk_rcv.src
;
; Author:       Chris Fogelklou
;               Applications Engineer
;               Scenix Semiconductor Inc.
;
; Revision:     1.02
;
; Date:         January 25, 1999.
;
; Part:         SX28AC rev. 2.5
;
; Freq:         50Mhz
;
; Compiled using Parallax SX-Key software v1.0
;
; Version:      1.02
;
; Program Description: This is simple software to demonstrate the concept of
;                       converting an incoming frequency to a data train. This
;                       program simply watches the incoming frequency and outputs
;                       a high or a low to the RS-232 TX pin. For this reason, the
;                       PC that the board is connected to must be set to the desired
;                       baud rate. Example: If a 300,N,8,1 settings are desired,
;                       simply set the PC's terminal program to these settings and
;                       the SX modem will act as a simple frequency-voltage converter,
;                       translating the incoming frequency (1300Hz and 2100Hz) to
;                       an output voltage which the computer will receive as a 300,N,8,1
;                       data packet. Once the modem is connected, simply press a key
;                       on the keyboard to get it to pick up and begin receiving.
;
; Revision History:  1.0  Drew out the FSK receive code from the rest of the code
;                       surrounding it in the old modem demo software and implemented

```

```

;           it as a module by itself (December 17, 1998)
;           1.01 Added more documentation to the software for web-posting.
;           (January 25, 1999)
;           1.02 Added support for rev 1.4 boards (2/23/99)
;
; INPUTS:
; FSK input on fsk_rx_pin (rb.1) (Signal must be amplified until clipping
;           to overcome Schmitt Trigger inputs.)
; RS-232 input on rx_pin (ra.1)
;
; OUTPUTS:
; Received RS-232 characters on tx_pin (ra.2)
; LED flashes while receiving on led_pin (rb.0)
;
; RESOURCES:
; Program memory: 67 Words
; Data memory:   3 bytes
; I/O Count:    3-5   (Some may be optional)
;
; *****
; Device Directives
; *****
device pins28,pages1,banks8           ; 28-pin device, 1 pages, 8 banks of RAM
device oschs,turbo,optionx,stackx     ; High speed oscillator, turbo mode,
                                       ; option register extend, 8-level stack
freq 50_000_000                       ; default run speed = 50MHz
ID 'SFSKRX10'                         ; Version = 1.01

reset reset_entry                     ; JUMP to reset_entry label on reset

; *****
; Watches (For Debug in SX_Key software V.1.0 +)
; *****

watch fsk_trans_count,8,udec ; counts the delay between transitions

; *****
; Macros (These only required on SX DTMF DEMO board)
; *****
; oldboard           ; Uncomment this if board is not marked rev 1.4 or uses
;                   ; a DIP SX package.
disable_o macro 0           ; This macro disables the output
  ifdef oldboard
    setb in_out             ; switch on the new modem boards.
  else
    clrb in_out
  endif

```

```

        clr     flags
        endm

;*****
; Equates for FSK reception
;*****
glitch_th      equ     20      ; The threshold which defines a glitch (small spike which should be ignored)
low_count_error_th  equ     40      ; The lowest count allowed for a high frequency
low_high_th    equ     95      ; The lowest count allowed for a low frequency
high_count_error_th equ     140     ; The highest count allowed for a low frequency

;*****
; Pin Definitions (These definitions are for SX DTMF DEMO boards)
;*****

rx_pin         equ     ra.1      ; RS-232 Input pin
tx_pin         equ     ra.2      ; RS-232 Output pin
in_out         equ     ra.3      ; Switches between output
                                   ; and input on SX DTMF DEMO boards.

led_pin        equ     rb.0      ; Flashes while characters are
                                   ; being received.
hook           equ     rb.4      ; Goes on/off-hook.

;*****
; Global Variables
;*****
        org     $8              ; Global registers

flags         ds      1
        fsk_rx_en    equ     flags.0 ; Enables the FSK receiver.

;*****
; Bank 0 Variables
;*****
        org     $10

fsk_receive_bank =      $

        fsk_trans_count    ds      1      ; This register counts the number of counts
                                   ; between transitions at the pin
        rb_past_state    ds      1      ; This register keeps track of the previous
                                   ; state of port RB, to watch for transitions

;*****
; Interrupt
        org     $0              ; The interrupt Service routine starts at location zero.

```

1. waveform  
2. waveform analysis

```

; With a retlw value of -163 and an oscillator frequency of 50MHz, this
; code runs every 3.26us.
;*****
        jnb     fsk_rx_en,fsk_rx_out  ; jump out if the FSK receiver is not enabled
;*****
FSK_RECEIVE
        bank    fsk_receive_bank      ; switch to fsk_receive_bank of RAM

        add     fsk_trans_count,#1     ; Regardless of what is going on, increment the
        snc     ?                       ; transition timer. These get cleared when a transition
        jmp     :error                 ; takes place.

        cjb     fsk_trans_count,#low_high_th,fsk_timer_out ; as soon as it takes longer than 95 counts
        setb    tx_pin                 ; to transition, this must be a low frequency
        clrb    LED_pin                ; If a high is being sent, clear the LED

;fsk_timer_out
        mov     w,rb
        and     w,#00000010           ; get the current state of rb.
        xor     w,rb_past_state       ; compare it with the previous state of the pin
        jz      fsk_rx_out            ; if there was no change, then jump out, there is nothing to do.

; Now it is time to determine if the transition that took place indicates a bit was
received
        ; (it must be within some thresholds... below 20, ignore it, below 40, what???,
        ; below 95, high frequency, below 140, low frequency (already set), above 140,
        ; what???)

        cjb     fsk_trans_count,#glitch_th,:glitch_so_ignore ; pulse was below specs, ignore it... probably noise
        cjb     fsk_trans_count,#low_count_error_th,:error    ; pulse was not a glitch but wasn't long enough to mean
anything... huh?
        cjb     fsk_trans_count,#low_high_th,:high_frequency  ; pulse was within specs for a high frequency...
        cjb     fsk_trans_count,#high_count_error_th,:fsk_receive_done ; pulse was within specs for a low frequency (don't
do anything)
        jmp     :error                                          ; pulse was too long to mean anything, so do nothing.

;high_frequency
        clrb    tx_pin
        setb    LED_pin
        jmp     :fsk_receive_done
; a high frequency corresponds to low data.
; set the LED to indicate a LOW is being sent.

:error
;----- PUT ERROR HANDLING CODE IN HERE -----

```

65.2µs  
20  
40  
130.4

100 = 456.4µs

309µs

1300 → 767  
72 384µs

756µs = 356µs + 416  
H1 LO

HF 2100  
476µs = 216 + 268µs  
H1 LO

2100Hz → 238µs  
H1  
LO

102  
416 606 0864

```

:fsk_receive_done
:clr      fsk_trans_count      ; clear the bit counter.
:glitch_so_ignore      ; don't clear the counter if the data was a glitch
mov      w,rb      ; save the new state of RB.
and      w,#%00000010
mov      rb_past_state,w

fsk_rx_out
;*****
:ISR_DONE
; This is the end of the interrupt service routine. Now load 163 into w and
; perform a retiw to interrupt 163 cycles from the start of this one.
; (3.26us@50MHz)
;*****
mov      w,#-163      ;1      ; interrupt 163 cycles after this interrupt
retiw    ;3      ; return from the interrupt
;*****
; End of the Interrupt Service Routine
;*****

;*****
reset_entry
; Program Starts Here on Power Up
;*****
mov      m,#$0c      ; Initialize SX.
mov      !rb,#%11111101      ; enable Schmidt trigger on rb1 (for FSK receive)

mov      m,$0f
mov      ra,#%0110      ; init ra
mov      !ra,#%0010      ; ra0-1 = input, ra2-3 = output
mov      rb,#%00000000      ; init rb
mov      !rb,#%11101110      ; rb1 = FSK input, rb0 = output for LED,rb5 = hook
clrb     hook      ; go on hook.
setb     led_pin      ; turn on LED
clr      flags      ; Clear all flags

disable_o

jb      rx_pin,$      ; until the user presses a key, loop indefinitely

setb     hook      ; pick up the line
setb     fsk_rx_en      ; enable the FSK receiver

mov      !option,#%00011111      ; enable wreg and rtcc interrupt

main      jmp      $      ; jump here forever (ISR does all the work)

```